National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

TECHNICAL MEMORANDUM 98/241
October 1998

IMPLEMENTATION OF MULTI-BLOCKING IN THE TRANSOM PSEUDO-COMPRESSIBILITY SOLVER

David Hally

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

102-03097/

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

IMPLEMENTATION OF MULTI-BLOCKING IN THE **TRANSOM** PSEUDO-COMPRESSIBILITY SOLVER

David Hally

October 1998

Approved by: J.L. Kennedy:
Head / Warship Signatures and Safety Section

TECHNICAL MEMORANDUM 98/241

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

Abstract

TRANSOM is a Reynolds-averaged Navier-Stokes solver being developed at DREA for calculating the flow around ships and submarines. TRANSOM will eventually allow the fluid region to be broken into blocks of different types (e.g. structured or unstructured) with different solvers being used on each block.

As an intermediate step, this document describes the implementation in TRANSOM of a pseudo-compressibility solver suitable for use on structured multi-block grids: i.e. grids in which each block is structured. It is restricted to the case in which the coordinate metrics are continuous across connected block boundaries. In this case, the algorithms for calculating and assembling the linear system used by the solver can be generalized quite easily from the algorithm for a single block.

Résumé

TRANSOM est un logiciel de calcul Navier-Stokes pondéré Reynolds que l'on développe au CRDA pour les calculs de l'écoulement autour des navires et des sous-marins. Eventuellement, TRANSOM permettra de diviser la région fluide en blocs de différents types (structurés et non structurés) en utilisant différents logiciels sur chaque bloc.

Comme une étape intermédiaire, ce document décrit la mise en oeuvre dans TRANSOM d'un logiciel de calcul de pseudo-compressibilité adapté pour les grilles multi-blocs structurés, c'est-à-dire les grilles dans lesquelles chaque bloc est structuré. Il est restreint au cas dans lequel les paramètres des coordonnées sont continus entre les limites de blocs connectés. Dans ce cas, les algorithmes de calcul et d'assemblage du système linéaire utilisés par le logiciel de calcul peuvent être généralisés facilement à partir de l'algorithme d'un bloc simple.

DREA Technical Memorandum TM/98/241

Implementation of Multi-Blocking in the TRANSOM Pseudo-Compressibility Solver

by

David Hally

Executive Summary

Background

The computer program TRANSOM is intended to solve a number of ship-related problems of importance to DND: for example, flows around submarines, vortex generation from propellers and control surfaces, bilge vortex generation, and prediction of roll damping coefficients. In particular, it will enhance our ability to predict the flow around a ship hull and into the propellers. Knowledge of this flow is crucial for the design of efficient, quiet propellers.

TRANSOM is a multi-block, multi-method Reynolds-averaged Navier Stokes solver; multi-block because the flow is divided into several distinct regions called blocks; multi-method because a different solution method can be used on each of the blocks. Unlike the methods currently in use at DREA, TRANSOM solves the full Navier-Stokes equations in conjunction with a turbulence model; hence, when completed, it will be able to solve the flow around complex geometries such as a fully-appended ship or submarine.

The current version of TRANSOM includes two different solution methods which can be used on the blocks. One uses the finite element method; it is suitable for use on unstructured blocks. The second, the SPC solver, uses a pseudo-compressibility method on structured blocks.

Principal Results

This memorandum describes the extension of the SPC solver to a multi-block solver. It can now be used to solve the flow in any of the following three situations:

1. Flow in a single isolated block. This is the case originally implemented.
2. Flow in a single block which is part of a larger multi-block structure. When several solvers are used concurrently to solve the flow in neighbouring blocks, one obtains a memory-efficient global solver because memory can be shared among each of the solvers. In general the price paid for low memory cost is slower convergence rates.
3. Flow in several connected blocks. In this case the solver creates a single global matrix to solve on each of the blocks simultaneously. This form of multi-block solver has good convergence characteristics but requires enough memory to store the global matrix.

Significance of Results

The inclusion of multi-block capability in TRANSOM is an important step in its evolution, allowing flows to be calculated over much more complex geometries than would otherwise be possible. The ability to run the solver in a memory-efficient mode is also an important requirement. Because the flows of interest to DREA are at very high Reynolds number (typically the order of 10^9 for the flow past a ship or submarine), a very large number of nodes will be necessary. For some problems, a memory-efficient mode of solution will be necessary in order to fit the problem into computer memory.

Future Work

For maximum flexibility of grid generation, two further extensions must be made. Currently, the slopes and spacing of grid lines must be continuous across block interfaces: that is, the coordinate metrics must be continuous. The multi-block structured solvers must be extended to handle the case of connected block boundaries with discontinuous metrics. This is being addressed under contract to the University of Toronto.

The second extension will define multi-block solvers that allow both structured and unstructured blocks in the same grid. This is a research topic which will be pursued in-house after TRANSOM has been fully converted to three-dimensional flows.

1 Introduction

TRANSOM is a Reynolds-averaged Navier Stokes solver being developed at DREA to calculate flows around ships and submarines. TRANSOM allows the fluid region to be broken into blocks of different types (e.g. structured or unstructured) with different solvers being used on each block. Reference 1 describes the overall design of TRANSOM.

The current version of TRANSOM includes two different solution methods which can be used on the blocks. The first is based on the program NSI2D developed for DREA under contract to the University of Toronto[2,3,4,5]; it is suitable for use on structured blocks. The algorithm is a variant of the pseudo-compressibility method developed by Rogers and Kwak[6]. Currently only two-dimensional flow can be modelled. Turbulence may be modelled using the Baldwin-Lomax model[7], the Chen-Patel variant of the k-ϵ model[8], or the Spalart-Allmaras model[9].

The second solution method is based on the finite element program MEF developed at Université Laval[10]; it is suitable for laminar flow on unstructured blocks. MEF was modified at DREA to model turbulence using a variant of the k-ϵ model[11] but this has not yet been incorporated into TRANSOM. Both two and three dimensional flow can be modelled.

Reference 12 describes the implementation of TRANSOM's structured pseudo-compressibility (SPC) solver on single structured blocks. The current document describes the extension of this solver for use on structured multi-block grids: i.e. grids in which every block is structured. It is restricted to the case in which the coordinate metrics are continuous across connected block boundaries. In this case, the algorithms for calculating and assembling the linear system used by the solver can be generalized quite easily from the algorithm for a single block.

There are three separate cases which must be addressed.

1. A solver on a single block in isolation. For any node in the block, all its neighbouring nodes also lie in that block. In this case a local numbering system may be used and the independent variables (e.g. pressure and velocity), the metrics, and the linear system (the left hand side matrix and the residual vector) need only be as large as the block. This is the case originally implemented and described in Reference 12.

2. A solver on a single block which is part of a larger multi-block structure. In this case, some nodes on boundaries of the block may have neighbours in different blocks. Therefore, the residuals are calculated using values of pressure, velocity, and the metrics from other blocks. A global numbering system must be used for the independent variables and the metrics. However, only values at the block nodes are updated, so the linear system is still constrained to the nodes of the block. Consequently, the linear system may use a local numbering system. Thus, the independent variables and metrics must be stored in large data-structures containing values over all global nodes, while the linear system may be stored in local data-structures whose size is only as large as the size of the block.

   In general, this type of solver will be included in a larger concurrent solver to find the solution over the whole block (see Reference 1 for a detailed description of concurrent solvers). In this form of solver, a linear system is assembled and solved on each block in turn. The memory required for the linear system can be reclaimed after it has been solved. Hence, the linear systems only require enough memory to solve the equations on

the largest block. Because the memory for the linear system is the major portion of the total memory requirement for each solver, the concurrent solution method is very memory efficient. On the other hand, because each block is decoupled from its neighbours, there will generally be a decrease in the convergence rate.

3. A solver which creates a single global matrix to solve several connected blocks simultaneously. In this case the independent variables, the metrics, and the linear system must all use global node numbering. This form of multi-block solver will have good convergence characteristics but will require enough memory to store the global matrix.

Wherever possible we want to re-use the existing single block code to implement each of these three cases. After making the modifications described in Section 3 it is possible to re-use almost all the single block code.

2 Structured Grids

The TRANSOM SPC solver calculates the flow on a structured block. Re-entrant boundaries (e.g. O or C topology grids) may be used by defining the connections between the boundary nodes. The multi-block pseudo-compressibility (MultiSPC) solver is an extension of the SPC solver which calculates the flow on a structured grid: a grid consisting of connected structured blocks. In this section, C++ classes that represent structured grids are described.

2.1 The ConnS2dBlock Class

The class ConnS2dBlock has been described in Reference 12, Section 4.3. In almost all respects the class, as described there, is adequate for use representing the connected blocks in a structured grid. The sole problem is the way in which the metrics are handled. In order to calculate the finite difference approximations of derivatives at a connected boundary node, it is often necessary to know the value of the metric at points which lie in the neighbouring block. This is possible if the metrics are stored in global arrays, as are the independent variables such as pressure and velocity. For this purpose ConnS2dBlock has been given an extra constructor which permits sharing of the metrics in global arrays:

```
ConnS2dBlock(unsigned m1, unsigned m2, const Node &f, unsigned maxn, Metric2d *m,
             double *d = 0);
```

Constructs a two-dimensional structured block with dimensions m1 × m2 starting at node f and using the storage at d. The metrics are shared with the metrics m starting at node n. The maximal number of nodes in the block is maxn.

2.2 The S2dGrid Class

An S2dGrid is a two-dimensional grid which contains only structured blocks. The node numbering for the grid is shared by all the blocks, so that the numbering for each block does not start at zero. S2dGrid also maintains global metrics.

The S2dGrid class is a specialization of the Grid class inheriting all its public and protected member functions: see Reference 1, Section 2.5, for a detailed description of the Grid class. In addition, it has following constructors and member functions.

S2dGrid(unsigned n);

Makes a structured two-dimensional grid having n nodes.

S2dGrid(unsigned n, const PointColl &p, const Node &f);

Makes a grid with n nodes sharing coordinates with p starting at node f. Grid numbering starts at zero. This constructor is used when the structured grid is embedded within a larger grid: i.e. it is treated as a block in a Grid. The coordinates are shared with the enclosing grid.

ConnS2dBlock* get_block(const Str &name) const;

Returns a pointer to block with the given name. Returns a null pointer if there is no block of that name.

S2dGrid maintains a list of all the connections between its blocks. Each connection is an instance of the struct S2dBlockConn:

```
struct S2dBlockConn
{
  int separated;                          // True if the connection is separated
  Str name1, name2;                       // The block names
  S2dNodeOrg::BndType bnd1, bnd2;         // The connected boundaries
  Index lo1, hi1, lo2, hi2;               // The ranges of connected nodes
};
```

(S2dNodeOrg::BndType is described in Reference 12, Section 4.1.2).

In a separated boundary connection, no nodes are shared by the two blocks. Rather, the nodes on bnd1 are the immediate neighbours of the nodes on bnd2, and vice versa. In a separated connection, each connected node is represented only once in the grid.

If the connection is not separated, two copies of each connected node are maintained, one belonging to first block, the other belonging to the second block. The standard O and C block topologies used by the classes OTopS2dBlock and CTopS2dBlock (see Reference 12, Sections 4.4 and 4.5) are examples of connections which are not separated.

The following S2dGrid member function returns an iterator over all the block connections.

QueueIter<S2dBlockConn> get_connections_iter() const;

The class S2dGridIter is an iterator over all the blocks in an S2dGrid returning a pointer to each of them in turn. The prototype for its constructor is:

S2dGridIter(const S2dGrid &g);

Standard TRANSOM iterator syntax is used for both these iterators: see Reference 1, Section 2.2.

A structured grid can be defined from records in an OFFSRF input file using an extractor overloaded for the Sgrid and OFFSRF_ifstream classes (see Reference 13 for a description of

OFFSRF files and the class OFFSRF_ifstream). If g is an SGrid and in is an OFFSRF_ifstream, then the code

```
in >> g;
```

will read the definition of the structured grid from the OFFSRF file associated with the input stream in. The records in the file should be in the format shown in Figure 1. The NAME, COORDINATES, STRUCTURED BLOCK, O BLOCK, and C BLOCK records are described in Reference 12, Section 4. The BOUNDARY CONNECTION record defines a connection between the boundaries of two blocks. Each of the two lines in the record defines the range of nodes on a single boundary of a given block. These nodes are shared by the two blocks.

The SEPARATED BOUNDARY CONNECTION record is similar except that the boundary connection is separated.

3 Concurrent Solvers on Connected Blocks

This section describes the changes that were necessary to enable the use of the SPC solver on a single block of a multi-block grid. The only difference between this solver and the solver on an isolated grid is that the boundary pointers can point to nodes which lie outside the block. Nevertheless, this relatively small difference has led to significant changes in implementation of the solver.

3.1 Return to Halo Points

The original TRANSOM structured pseudo-compressibility (SPC) solver was almost a direct implementation of the Fortran program NSI2D developed for DREA under contract to the University of Toronto. Most of the calculations were performed by NSI2D subroutines called directly from the TRANSOM C++ code. The methods used have been documented in References 1–6, 12, and 14. The calculations proceed by first calculating fluxes at the boundaries of the cell surrounding each node, then using the fluxes to assemble a linear system which is solved to determine an increment for the independent variables. To handle re-entrant boundaries in O and C topologies, the earliest version of NSI2D used topology specific systems of halo points[2]: i.e. the block was enlarged to include extra points, the halo points, lying in neighbouring blocks. In this scheme arrays containing values of properties at each node (position, velocity, etc.) must also be enlarged. Subsequent versions of NSI2D discarded the halo points in favour of a system of pointers to neighbouring nodes and flux boundaries. In this way the flux at a cell boundary never needed to be calculated twice, arrays were smaller, and the dependence on topology was removed. Unfortunately the flux boundary pointers (these are the pointers stored in the array bndfluxmap: see Reference 3) are not well suited to a solver on a single block within a larger multi-block grid. The problem is that the calculation follows the flux pointers to cell boundaries in another block *and expects to find flux values for those cell boundaries.* However, in general, no fluxes will have been calculated for those cell boundaries. Therefore, it is necessary to return partially to the concept of halo points. In the scheme now implemented, the node boundary pointers (array bndmap: see Reference 3) are retained as are the boundary inversion flags (array

{STRUCTURED GRID: *ndim npts* [*start-node*]
  {NAME: *name* }
  {COORDINATES
    x_0 y_0 z_0
    x_1 y_1 z_1
    ⋮ ⋮ ⋮
    x_n y_n z_n
  }COORDINATES
  {OUTPUT FILE: *file-name* }

  {STRUCTURED BLOCK: m_1 m_2 [*start-node*]
    < Records to define a structured block: see Reference 12, Section 4.3 >
  }STRUCTURED BLOCK

  {O BLOCK: m_1 m_2 [*start-node*]
    < Records to define an O-block: see Reference 12, Section 4.4 >
  }O BLOCK

  {C BLOCK: m_1 m_2 *itail1 itail2* [*start-node*]
    < Records to define a C-block: see Reference 12, Section 4.5 >
  }C BLOCK

  {BOUNDARY CONNECTION
    block-name-1 boundary-name-1 lo1 hi1
    block-name-2 boundary-name-2 lo2 hi2
    {CONTINUOUS} or {DISCONTINUOUS}
  }BOUNDARY CONNECTION

  {SEPARATED BOUNDARY CONNECTION
    block-name-1 boundary-name-1 lo1 hi1
    block-name-2 boundary-name-2 lo2 hi2
    {CONTINUOUS} or {DISCONTINUOUS}
  }SEPARATED BOUNDARY CONNECTION
}STRUCTURED BLOCK

Figure 1: *Structure of the STRUCTURED GRID input record.*

bndinv); they contain all the necessary information about block connections. Armed with the node pointers, it is not necessary to use separate halo arrays for the independent variables (pressure, velocity, etc.) or for the metrics, since they have known values at all nodes of the grid. Their values at the halo points can be obtained using the node pointers. On the other hand, halo arrays (e.g. arrays whose index range is extended from $[0, n-1]$ to $[-3, n+2]$) are used for all the temporary arrays used in the calculations, including all the arrays which store fluxes. Fluxes are calculated explicitly at all interior and halo flux boundaries. The boundary flux pointers are no longer necessary.

3.2 Reduction of Temporary Arrays to One Dimension

Reference 3 describes in detail how the calculation of the linear system can be reduced to the independent calculation of fluxes in each of the coordinate directions. This calculation can be reduced further to the successive calculation of the fluxes along each row of nodes. Thus, one should be able to calculate the fluxes along one row, assemble them into the linear system, then proceed to the next row. Storage arrays need only be large enough to contain the fluxes for a single row of nodes. When boundary flux pointers were used, the difficulty was that the flux pointers might point to flux boundaries in a different row: for example, this would occur along the wake cut of a C-topology block. Therefore the fluxes had to be calculated and stored at all nodes prior to assembling. With halo points this is no longer a problem since the fluxes at the halo points are calculated explicitly from values of the independent variables at the halo nodes. Hence, all the temporary arrays used in the calculation may be reduced from two-dimensional arrays for the whole block, to one-dimensional array for a single row of nodes. The memory requirements for the solver are reduced significantly.

3.3 Global Numbering

The ability of the boundary node pointers to point at nodes outside the block means that the local node numbering of the block cannot be used. Instead a global numbering system for the whole multi-block grid must be used. In addition, the values of the independent variables and the metrics for the whole grid must be made available so that their values at nodes outside the grid may be obtained. This has required some minor changes in the NSI2D subroutines used by TRANSOM. In particular, the number of the first node in the block must be passed to many of the subroutines for use as an offset between the local numbering and the global numbering.

3.4 Independent Boundary Pointers in the Matrix Representation

In NSI2D the boundary node pointers are used not only when assembling the linear system, but also to help give it its structure. For a point on a boundary, the node pointer is used to determine which column in the matrix corresponds to the neighbouring node across the boundary. However, in the case of a single block solver within a multi-block grid, this will no longer work. The column for the node across the boundary does not exist. Thus, the pointers into neighbouring blocks are necessary when the residual is calculated, but they must be ignored by the matrix representation. In the original modification of TRANSOM to handle this case, a

separate set of boundary pointers was introduced. One set of pointers described the connections of nodes between blocks; the other set described the structure of the matrix to be solved.

More recently the representation of sparse matrices and linear systems in TRANSOM has been completely revised and generalized[15]. The new matrix representations completely separate the representation of the matrix from any structure of the block. Thus the sparse matrices can be used for both single block and multi-block solvers on structured or unstructured grids.

3.5 Problems with the Current Implementation

In the current implementation, all the information about the connection of two blocks is stored in the boundary node pointers (bndmap) and the boundary inversion flags (bndinv). However, these are insufficient in the case when a ξ-boundary connects with an η-boundary (ξ and η are the block coordinates: see Reference 12). In this case the metrics must be transposed at the halo points since a derivative by ξ becomes a derivative by η and vice versa. Therefore, at present, ξ-boundaries must be connected with ξ-boundaries and η-boundaries with η-boundaries. The simplest solution to this problem is to treat the metrics as temporary variables which are recalculated on each sweep. They will then be stored in one-dimensional halo arrays saving significant storage space. The penalty is a very small decrease in efficiency.

There is also a problem when four blocks meet at a single node. Suppose this node is the first node of the block: $(\xi, \eta) = (0, 0)$. Consider the calculation of the viscous fluxes in the ξ-direction at the halo points beyond this node. Derivatives by ξ are easily calculated using the halo points, but there is a contribution to the flux from the η-derivatives of the independent variables. To calculate these derivatives we need to use the boundary pointers on $\eta = 0$ to get the points which are equivalent to $\eta = -1$, but these pointers do not extend to the halo points. The solution is to extend haloing to the boundary node pointers which specify the block connections: i.e. the pointers stored in the class ConnS2dBlock.

4 The Full Multi-Block Solver

With the changes made to implement the single block solver in a multi-block grid, and the changes made to the representation of sparse matrices, the implementation of a full multi-block solver with a global matrix is straightforward. The global matrix represents the influence of each node in the grid on every other node. The right hand side of the linear system is represented as a vector over all the nodes in the grid. The multi-block solver is represented as a list of sub-solvers: one for each block. The global matrix is passed to each sub-solver when it is constructed. The sub-solvers are used to assemble the linear system into the global matrix and the global residual vector. A global node numbering system is used on each block so that appropriate elements in the global matrix are easily found. For nodes which are shared by two different blocks, one is chosen as a master, the other a slave. The equations for the master node are assembled normally while those for the slave simply require that the independent variables at the slave node be the same as those for the master node. The linear system is solved using one of the sparse linear system solvers described in Reference 15. The solution vector is used to update the independent variables.

Figure 2: *Geometry for the flow over a backward-facing step. Vertical dimensions scaled by a factor of two.*

The linear system is represented by the class NodeMapSpLinSys described in Reference 15. It allows the user to choose, via the TRANSOM input file, a solution method for the linear system, a preconditioner for the matrix, left or right preconditioning, and a method of node re-ordering. The solution method can be one of the Successive Substitution method (not recommended), the Conjugate Gradient Squared (CGS) method, the Quasi-Minimum Residual (QMR) method, or the Generalized Minimum Residual (GMRES) method. Either diagonal or incomplete LU factorization of level zero[16] (ILU0) preconditioning can be chosen. Preconditioning can be either on the left or right. If desired (but not recommended) preconditioning can also be ignored. The nodes (and hence the elements of the matrix) can be re-ordered in five ways: no re-ordering, reverse ordering, random ordering, ordering using the Cuthill-McKee algorithm[17], or ordering using the reverse Cuthill-McKee algorithm. It is also possible to choose, from the five options, the ordering which minimizes the mean bandwidth of the matrix. Details on all of these methods are available in References 15 and 16.

5 Examples

In this section examples of two simple multi-block flows are described: flow over a backward-facing step and flow past a square obstacle.

5.1 Flow Over a Backward Facing Step

In this section TRANSOM calculations of flow over a backward-facing step are described. The geometry of the flow is illustrated in Figure 2 and a portion of the grid near the step in Figure 3. The channel is 20 units long with the step 4 units from the inlet. Upstream of the step the channel is 0.5 units wide. Downstream of the step it is one unit wide. The flow region was split into two blocks: one upstream of the step and the other downstream of the step. A 'separated connection' was used to join the blocks so that they shared no nodes.

Figure 4 shows the results of a TRANSOM run at a Reynolds number of 100 based on channel width downstream of the step. The flow is laminar. At the inlet, a quadratic velocity profile was specified. At the outlet, the pressure was set to a uniform value of zero. The flow was initialized to uniform flow along the channel with pressure equal to zero everywhere.

The flow was calculated in two ways: in the first, a full multi-block solver was used; in the second, separate Pressure/Velocity solvers on each block were run concurrently. The latter case will be called the disjoint solver. All solution parameters were identical for the two runs. The two solutions match to within the convergence tolerances.

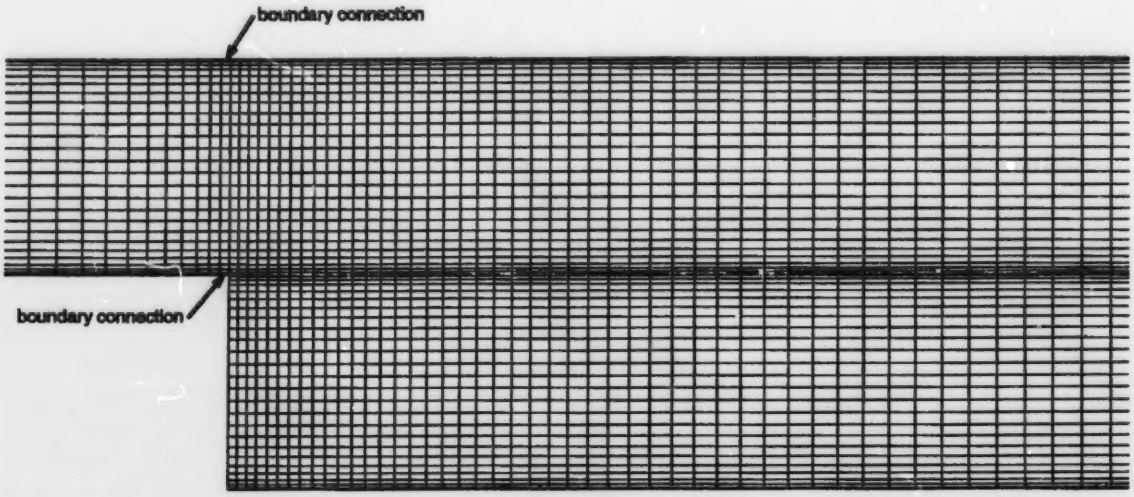

Figure 3: *The grid used for the flow over a backward-facing step. Only the portion near the step is shown.*

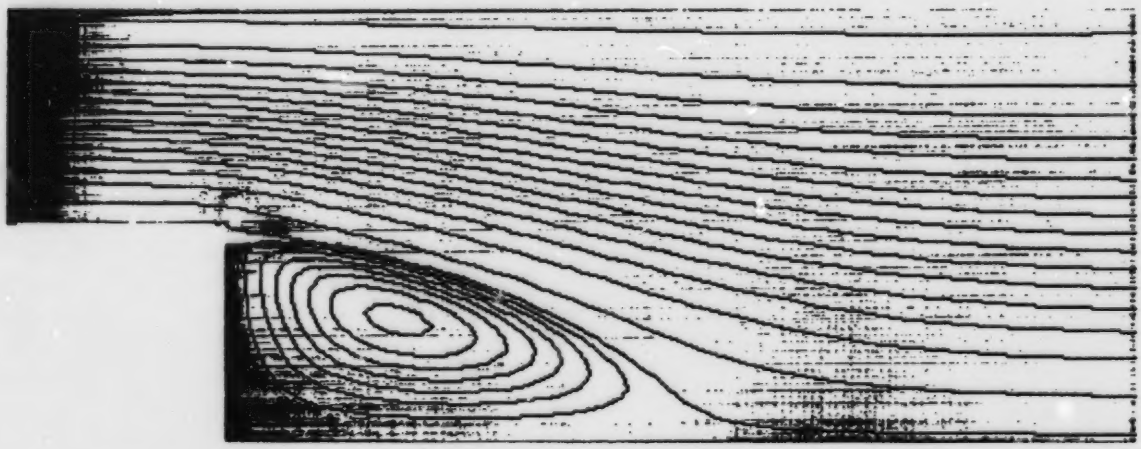

Figure 4: *Coloured pressure distribution and velocity contours for the flow over a backward-facing step at* $R_e = 100$.

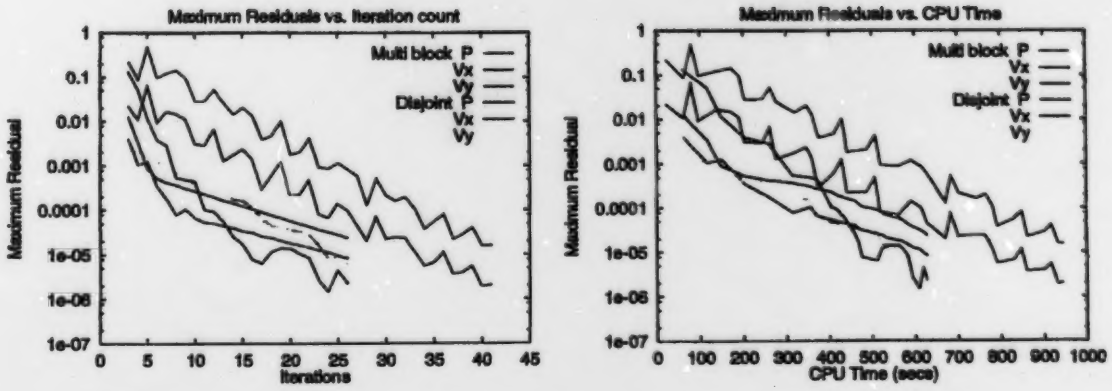

Figure 5: *Convergence histories for the flow over a backward-facing step at $R_e = 100$. The curves labelled 'Multi block' show the maximum residuals for the multi-block Pressure/Velocity solver. The curves labelled 'Disjoint' show the maximum residuals for the separate Pressure/Velocity solvers run concurrently.*

The convergence criterion was that the maximum residual be less than 10^{-4} and also that the maximum difference between independent variables at any node also be less than 10^{-4}. A similar convergence criterion was used for all the runs reported in this memorandum.

Convergence histories for the two runs are compared in Figure 5. Only the residual history is shown; the history of the differences in independent variables is very similar. It is interesting to note that the convergence rate for the two cases is very similar except for the first few iterations. Moreover, the CPU time per iteration is lower for the disjoint solver because the matrices used by the disjoint solver are better conditioned than the global matrix used by the multi-block solver, and hence, fewer iterations are required by the GMRES matrix solver. Therefore, the time penalty for using the disjoint solver is not too severe in this case.

Figure 6 shows similar TRANSOM solutions but at a Reynolds number of one million. The flow is fully turbulent. The Spalart-Allmaras (SA) turbulence model was used. At the inlet, a fully-developed channel flow velocity profile was specified. At the outlet, the pressure was set to a uniform value of zero. At both the inlet and the outlet the Spalart-Allmaras viscosity, $\tilde{\nu}$ was calculated by extrapolation from the interior of the flow. The flow was initialised to uniform flow along the channel with pressure equal to zero everywhere.

Once again the flow was calculated using full multi-block and disjoint solvers. In the former, multi-block pressure/velocity and Spalart-Allmaras sweeps were run concurrently. In the latter two disjoint pressure-velocity sweeps were run concurrently with two disjoint Spalart-Allmaras sweeps. All solution parameters were identical for the two runs. The two solutions match to within the convergence tolerances.

Convergence histories for the two runs are compared in Figure 7. Only residuals for the pressure and the Spalart-Allmaras viscosity are shown. The convergence histories for the velocity are similar. It was the size of the differences in the independent variables which was the limiting

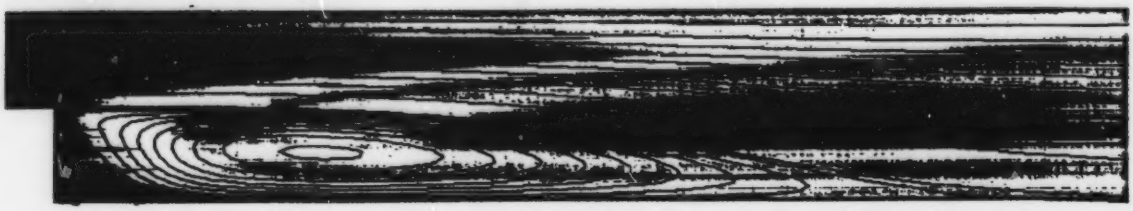

Figure 6: *Coloured distribution of turbulent viscosity and velocity contours for the flow over a backward-facing step at* $R_e = 10^6$.

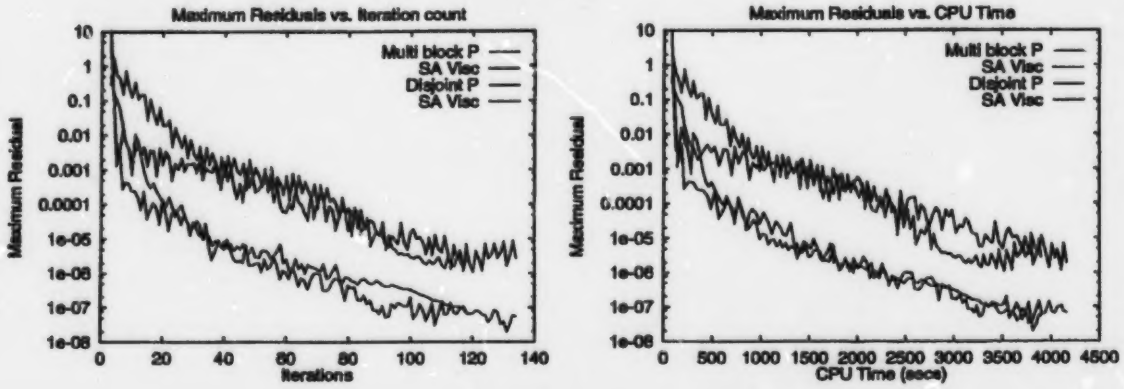

Figure 7: *Convergence histories for the flow over a backward-facing step at* $R_e = 10^6$. *The curves labelled 'Multi block' show the maximum residuals for the full multi-block solver. The curves labelled 'Disjoint' show the maximum residuals for the separate PV and SA solvers run concurrently.*

factor for convergence; hence, convergence was attained even though the residuals had risen somewhat over the final few iterations.

For this case the disjoint solver actually converged slightly before the multi-block solver. More iterations were required for the disjoint solver but the time per iteration was sufficiently lower that the overall execution time was less than that of the multi-block solver.

5.2 Flow Past a Square Obstacle

In this section, TRANSOM calculations of flow past a square obstacle step are described. The geometry of the flow is illustrated in Figure 8. The square has a side of length 1.0 and the flow is calculated in a larger square with a side of length 41.0. The flow region is split into four blocks of similar shape, though blocks 2 and 4 are rotated 90° relative to blocks 1 and 3. A 'separated connection' was used to join the blocks so that they shared no nodes. At the left and lower boundaries the flow is a uniform stream with an angle of attack of 30°. At the right and upper boundaries the pressure is set to zero. The flow was initialized to uniform flow equal to the free stream with pressure equal to zero everywhere.

As with the flow over a backward-facing step, the flow was calculated at Reynolds numbers of one hundred and one million using both full multi-block and disjoint solvers. At the higher Reynolds number the Spalart-Allmaras turbulence model was used. For each Reynolds number, the full multi-block and disjoint solutions match to within the convergence tolerances. Figures 9 and 10 show the results of the TRANSOM runs at each Reynolds number.

Convergence histories for the four runs are compared in Figure 11. Unlike the flow over a backward-facing step, for this case the disjoint solvers converge significantly more slowly than the full multi-block solvers. For laminar flow, the full multi-block solver used 13% less time per iteration than the disjoint solver. For the turbulent flow this trend was reversed, the disjoint solver using 22% less time per iteration than the full multi-block solver.

After about 200 iterations, there is a decrease in the rate of convergence of the disjoint solver in the high Reynolds number case. The cause is not yet known.

6 Concluding Remarks

The inclusion of multi-block capability in TRANSOM is an important step in its evolution, allowing flows to be calculated over much more complex geometries than would otherwise be possible. For maximum flexibility of grid generation, two further extensions must be made. First, the multi-block structured solvers must be extended to handle the case of connected block boundaries with discontinuous metrics. This is being addressed under contract to the University of Toronto. Second, multi-block solvers must be defined that allow both structured and unstructured blocks in the same grid. This is a research topic which will be pursued in-house after TRANSOM has been fully converted to three-dimensional flows.

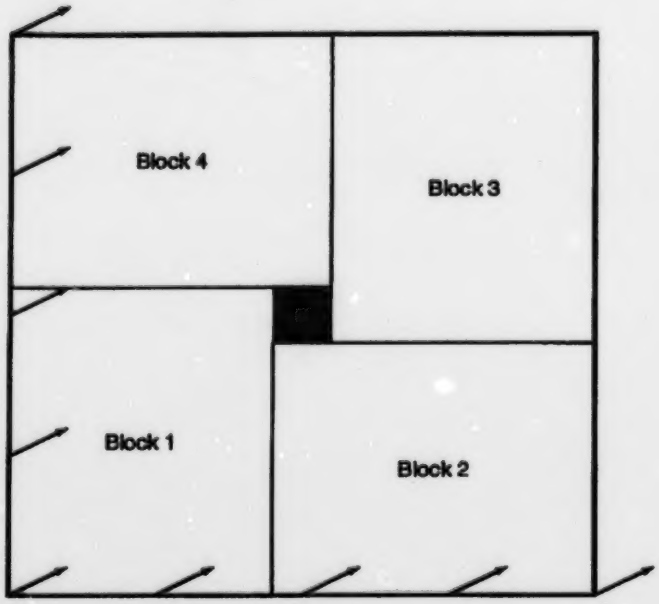

Figure 8: *Geometry for the flow past a square obstacle (not to scale).*

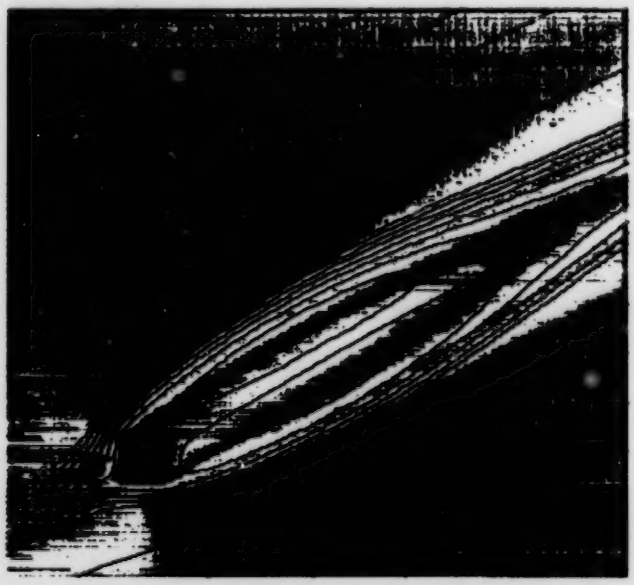

Figure 9: *Coloured pressure distribution and velocity contours for the flow past a square obstacle at* $R_e = 100$.

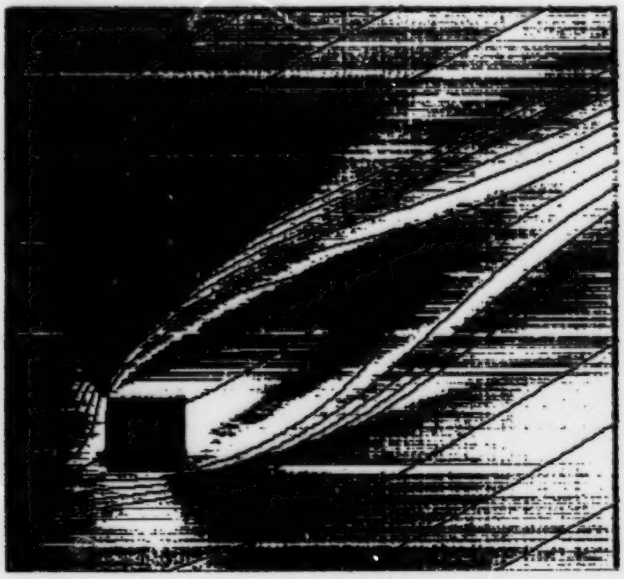

Figure 10: *Velocity contours and coloured distribution of velocity magnitude for the flow past a square obstacle at* $R_e = 10^6$.

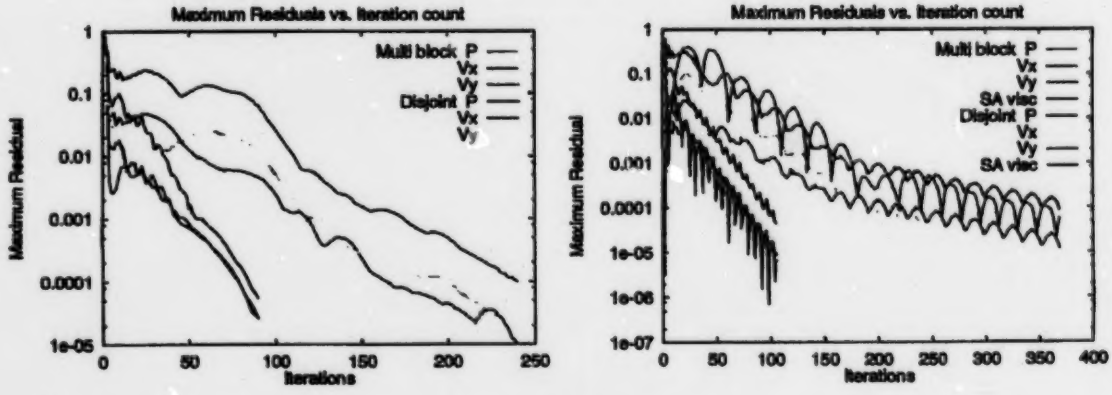

Figure 11: *Convergence histories for the flow past a square obstacle at* $R_e = 100$ *(on the left) and* $R_e = 10^6$ *(pn the right). The curves labelled 'Multi block' show the maximum residuals for the full multi-block solvers. The curves labelled 'Disjoint' show the maximum residuals for the disjoint solver.*

References

1. D. Hally, "TRANSOM: A Multi-method Navier-Stokes Solver: Overall Design," DREA Technical Memorandum 97/231, 1997.

2. D. J. Hall, D. W. Zingg, and C. R. Ethier, "A Laminar Incompressible Navier-Stokes Flow Solver," DREA Contractor Report CR/93/462, 1993.

3. D. Hally, "Revisions to NSI2D," DREA Technical Communication 94/303, 1994.

4. D. J. Hall, D. W. Zingg, and C. R. Ethier, "A Two-Dimensional Incompressible Navier-Stokes Turbulent Flow Solver," DREA Contractor Report CR/94/435, 1994.

5. P.D. Giansante, D. W. Zingg, and C. R. Ethier, "A Two-Dimensional Incompressible Navier-Stokes Turbulent Flow Solver: Code Enhancement and Convergence Acceleration," DREA Contractor Report CR/95/490, 1995.

6. S. E. Rogers and D. Kwak, "An Upwind Differencing Scheme for the Incompressible Navier-Stokes Equations," NASA, Technical Report TN 101051, Nov. 1988.

7. B. Baldwin and H. Lomax, "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78–257, AIAA 16^{th} Aerospace Sciences Meeting, Reno, Nevada, Jan. 1978.

8. H. C. Chen and V. C. Patel, "Near-Wall Turbulence Models for Complex Flows Including Separation," *AIAA Journal* **26**(6), 641–648 (1988).

9. P. R. Spalart and S. R. Allmaras, "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Technical Report, 1992.

10. G. Dhatt, G. Touzot, and G. Cantin, *The Finite Element Method Displayed* (John Wiley and Sons, New York, 1984). ISBN 0 471 90110 5.

11. F. Pineau, "A Finite Element Based Reynolds Averaged Navier-Stokes Solver for Modelling Three-Dimensional Turbulent Flows," DREA Report 93/106, 1993.

12. D. Hally, "Implementation of a Pseudo-Compressibility Navier-Stokes Solver in TRANSOM," DREA Technical Memorandum 98/239, 1998.

13. D. Hally, "C++ Classes for Reading and Writing files in OFFSRF Format," DREA Technical Communication 94/302, 1994.

14. M. Blanco, D. W. Zingg, and C. R. Ethier, "A Three-Dimensional Incompressible Navier-Stokes Solver," DREA Contractor Report in review, 1997.

15. D. Hally, "C++ Classes for Sparse Matrices and Iterative Linear System Solvers," DREA Technical Memorandum 98/240, 1998.

16. R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donatao, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Available from `templates@cs.utk.edu`.

17. E. H. Cuthill, "Several Strategies for Reducing the Bandwidth of Matrices," in *Sparse Matrices and their Applications*, edited by D. J. Rose and R. A. Willoughby (Plenum Press, New York, 1972), pp. 157–166.

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Atlantic
P.O. Box 1012
Dartmouth, N.S. B2Y 3Z7

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

Unclassified

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

Implementation of Multi-Blocking in the TRANSOM Pseudo-Compressibility Solver

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Hally, David

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO OF PAGES (total containing information include Annexes, Appendices, etc).	6b. NO. OF REFS (total cited in document)
October 1998	22	17

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

DREA Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address).

Defence Research Establishment Atlantic
P.O. Box 1012,
Dartmouth, N.S. B2Y 3Z7

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant).	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written).
1.g.a	

10a ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document).	10b OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor).
DREA TM/98/241	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected).

Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

TRANSOM is a Reynolds-averaged Navier-Stokes solver being developed at DREA for calculating the flow around ships and submarines. TRANSOM will eventually allow the fluid region to be broken into blocks of different types (e.g. structured or unstructured) with different solvers being used on each block.

As an intermediate step, this document describes the implementation in TRANSOM of a pseudo-compressibility solver suitable for use on structured multi-block grids: i.e. grids in which each block is structured. It is restricted to the case in which the coordinate metrics are continuous across connected block boundaries. In this case, the algorithms for calculating and assembling the linear system used by the solver can be generalized quite easily from the algorithm for a single block.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

Fluid Flow
Turbulence
Reynolds-Averaged Navier-Stokes equations
Hydrodynamics
Computer Programs
TRANSOM